# ПАРАЛЛЕЛЬНО-ПОСЛЕДОВАТЕЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ ТРЕУГОЛЬНЫХ СИСТЕМ НА ПРОЦЕССОРНОМ КОЛЬЦЕ

*Димитрий Львович Головашкин[1] (старший научный сотрудник, e-mail: dimitriy@smr.ru),*
*Надежда Николаевна Журавлёва[2] (студентка, e-mail: juravleva@mail.ru)*
*[1] - Институт систем обработки изображений РАН,*
*[2] - Самарский государственный аэрокосмический университет им. С.П. Королева*

***Аннотация***

Работа посвящена улучшению известного параллельного алгоритма решения треугольных систем на кольце. Предложенный подход основывается на последовательном исполнении заключительной части алгоритма, характеризующейся низкой эффективностью. Уместность разработанного приема подтверждена результатами вычислительных экспериментов.

*Ключевые слова:* матрица треугольного вида, параллельный алгоритм, процессорное кольцо.

## *Введение*

Широкое применение вычислительного эксперимента в научной практике и развитие микропроцессорной архитектуры обуславливают постоянный интерес как к разработке новых численных методов, так и к совершенствованию хорошо известных. Не будет преувеличением отметить задачу решения систем линейных алгебраических уравнений (СЛАУ) как наиболее массовую [1]. Ей посвящены главы широко известных учебников, рассчитанных на читателей без математического образования [2] и на читателей - специалистов [3]. В компьютерной оптике решение СЛАУ является заключительной частью таких разных подходов к моделированию прохождения электромагнитного излучения через дифракционные оптические элементы, как модовые методы, метод связанных волн, разностного решения уравнений Максвелла, методов конечных и граничных элементов решения уравнения Гельмгольца [4].

Вместе с тем, треугольным системам, которым посвящена настоящая работа, в массовой учебной литературе не уделяется специального внимания. Объяснением тому служит очевидность методики нахождения решения таких систем в скалярном случае. При векторизации вычислений рассматриваемая задача несколько усложняется: разработчик оказывается перед выбором строчно-ориентированного алгоритма (основанного на скалярном произведении) либо столбцово-ориентированного (опирающегося на saxpy) [1]. Переход к параллельным вычислениям приводит к алгоритму [5], уже превышающему по сложности LU разложение [6]. В отличие от последнего, характеризуемого для СЛАУ размерности n объемом арифметических операций $O(n^3)$ и количеством коммуникаций O(n), ставший классическим алгоритм из [5] связан с $O(n^2)$ арифметическими операциями при том же количестве коммуникаций. Доля длительности арифметических операций в общем времени вычислений, а следовательно, и эффективность [7] у алгоритма решения треугольной системы оказывается много ниже по сравнению с параллельным алгоритмом LU разложения. При том, что и тот, и другой используются для решения одной СЛАУ. Указанное несоответствие обуславливает необходимость повышения эффективности параллельного алгоритма из [5], чему и посвящена предлагаемая работа.

Известное утверждение о неактуальности синтеза параллельных алгоритмов решения треугольных СЛАУ (цитируемое в [1]) в силу меньшего количества арифметических операций по сравнению с LU-разложением представляется авторам необоснованным. Например, в работе [8], посвященной разреженным СЛАУ большой размерности, особо указывается на параллельное решение треугольных систем, как на «камень преткновения» при реализации метода неполной факторизации в модификации Айзенштата.

Приведенный параллельно-последовательный алгоритм основан на комбинации векторного варианта решения треугольной СЛАУ из [1] и параллельного варианта из [5]. Поэтому первым будет рассмотрен векторный алгоритм, за ним следует параллельный, завершает изложение авторский вариант метода.

## *1. Векторный столбцово-ориентированный алгоритм решения треугольных систем*

Выбор в настоящей работе столбцово-ориентированного алгоритма решения СЛАУ треугольного вида согласуется с общей ориентацией на столбцово-ориентированные алгоритмы, принятой в [1]. Такое предпочтение объясняется не только традиционным использованием в матричных вычислениях языка программирования Фортран, но и современной практикой. Например, в библиотеке CUBLAS версии 2.2 (появившейся в мае 2009 года) для варианта языка Си (обычно подразумевающего хранение матриц по строкам), используемого при кодировании векторных вычислений на видеокартах фирмы NVIDIA, предполагается хранение двумерных массивов по столбцам [8].

Для определенности здесь и далее будем рассматривать систему

$$Lx = b, \qquad (1)$$

где матрица $L \in \mathbb{R}^{n\times n}$ нижняя унитреугольная; $x, b \in \mathbb{R}^{n\times 1}$. Тогда традиционный столбцово-ориентированный алгоритм из [1], записанный в векторной нотации, основанный на операции saxpy и использующий замещение вектора правой части b

вектором неизвестных x, выглядит следующим образом.

Алгоритм 1

**for** j=1:n – 1 {проход по столбцам матрицы L}

{модификация правой части}

b(j+1:n)=b(j+1:n) – b(j)·L(j+1:n,j);

**end**

К сожалению, как показано в [1], приведенный лаконичный вариант не может послужить основой для синтеза параллельного алгоритма. Необходима модификация, предложенная там же.

Алгоритм 2

j=0; z(1:n)=0;

**while** j<n {проход по столбцам матрицы L}

{нахождение j-ой компоненты вектора x}

j=j+1; b(j)=b(j) – z(j);

{модификация вектора z}

z(j+1:n)=z(j+1:n)+L(j+1:n, j)·b(j);

**end**

В отличие от Алгоритма 1, в данном введен вектор $z \in \mathbb{R}^{n\times 1}$, модифицируемый посредством операции saxpy и накапливающий значения найденных неизвестных b(j), подставленных в уравнения системы (1).

### *2. Параллельный алгоритм решения треугольных систем на кольце*

Авторы [1,5] производят циклическую декомпозицию L и b, учитывая треугольную структуру матрицы системы и избавляя алгоритм от простоев. Перед началом вычислений в локальной памяти процессора с номерном μ ($1\le\mu\le p$, где p – количество процессоров) хранится матрица $L_{loc}=L(1:n, \mu: p: n)$ и столбец $b_{loc}=b(\mu: p: n)$. Для простоты в [1] принимается $n/p \in \mathbb{N}$. Левый и правый соседи процессора по кольцу обозначаются как left и right; вектор, пересылаемый по кольцу, как w.

Вывод алгоритма начинается в [1,5] с представления j-ой компоненты вектора x, относящейся к ведению процессора μ, через скалярное произведение фрагмента j-ой строки матрицы L и уже найденных компонент столбца x.

$$x(j)=b(j)-\sum_{i=1}^{j-1}L(j,i)x(i)= \\ =b(j)-\sum_{i=1}^{\min\{p,j-1\}}L(j,i:p:j-1)x(i:p:j-1) \quad . \qquad (2)$$

С учетом того, что каждое слагаемое второй суммы хранится в памяти одного процессора, обозначим

$$L(j,i:p:j-1)x(i:p:j-1)=\begin{cases} z_i^{(k)}(j), i=1:\mu-1 \\ z_i^{(k-1)}(j), i=\mu:p \end{cases}, \qquad (3)$$

где

$$z_\lambda^{(k)}=L(:,\lambda:p:\lambda+kp)\,x(\lambda:p:\lambda+kp) \qquad (4)$$

вектор z, сформированный процессором λ на k–ом обороте по кольцу вектора w. Подставляя (3) в (2), получим:

$$x(j)=b(j)-\sum_{i=1}^{\mu-1}z_i^{(k)}(j)-z_\mu^{(k-1)}(j)-\sum_{i=\mu+1}^{p}z_i^{(k-1)}(j). \qquad (5)$$

Пусть μ-ая компонента вектора w ($w\in\mathbb{R}^{p\times 1}$), обращающегося по кольцу, формируется как

$$w(\mu)=\sum_{i=1}^{\mu-1}z_i^{(k)}(j)+\sum_{i=\mu+1}^{p}z_i^{(k-1)}(j). \qquad (6)$$

Тогда расчет j-ой компоненты вектора неизвестных производится с учетом (5) и (6) по формуле

$$x(j)=b(j)-w(\mu)-z_\mu^{(k-1)}(j). \qquad (7)$$

Следуя [1], запишем параллельный алгоритм решения (1) на кольце.

Алгоритм 3

r=n/p; {число оборотов вектора w по кольцу}

col= μ:p:n; {вектор, ставящий в соответствие локальные и глобальные индексы}

k=0; {счетчик оборотов по кольцу}

z(1:n)=0; w(1:p)=0; {задание векторов z и w}

**while** k < r {вектор w сделает r оборотов по кольцу}

**If** k ≠ 0 or μ >1

recv(w, left); {первый оборот для первого процессора не сопровождается получением вектора w от левого соседа}

**end**

k=k+1; j=col(k);

$b_{loc}(k)=b_{loc}(k) – z(j) – w(\mu)$; {реализация ф. (7)}

**If** j < n {если вычисления не завершились}

{модификация части z, необходимой для формирования w}

q=min(n,j + p – 1);

$z(j+1: q) = z(j+1: q)+ L_{loc}(j+1: q, k)\cdot b_{loc}(k)$; { реализация ф. (4)}

{модификация циркулирующего вектора w в соответствии с ф. (6)}

**if** k < r {если оборот не последний и верхняя часть w еще понадобится}

$w(1: \mu – 1)= w(1: \mu – 1)+ z(j + p – \mu + 1 : j +p – 1)$;

$w(\mu) = 0$;

**end**

$w(\mu + 1: p)= w(\mu + 1: p)+z(j + 1 : j + p – \mu)$;

send(w, right); {пересылка w правому соседу по кольцу}

```
            {модификация остальной части z по ф. (4)}
            if  q < n
                        z(q +1: n)= z(q +1 : n)+ L_loc(q + 1: n, k)·b_loc(k)
            end
        end
end
```

Отметим, что каждый процессор хранит и модифицирует свой вектор z, в то время как вектор w формируется сообща.

Авторы настоящей работы реализовали приведенный алгоритм на языке Фортран 77 с применением библиотеки MPI, разворачивая векторные участки алгоритма в циклические конструкции. Расчеты, результаты которых представлены в табл. 1, производились на учебном кластере СГАУ, объединяющем ЭВМ с процессорами AMD Sempron (тактовая частота 1000 MHz), ОЗУ 1 Gb, работающими под управлением операционной системы Linux. Коммуникации между компьютерами осуществлялись посредством сети Ethernet (100 Mb).

*Таблица 1. Ускорение (S) двухпроцессорного параллельного алгоритма*

| n/1000 | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| S | 0,94 | 0,83 | 1,04 | 1,06 | 1,15 | 1,21 | 1,166 | 1,22 | 1,28 | 1,37 | 1,34 |

Отметим, что ускорение достигается лишь для значений n≥12000, в то время как для получения ускорения при реализации LU разложения на данном кластере достаточно систем размерностью на два порядка меньше. Даже в предельном рассмотренном случае n=20000 (матрицы большего размера не умещались в ОЗУ) получена весьма скромная для двухпроцессорной вычислительной системы величина S=1,34 (табл. 1). Причиной тому является отмеченная во введении невысокая эффективность Алгоритма 3. С увеличением числа задействованных в расчетах процессоров картина лишь усугубляется: при n=20000 трехпроцессорный алгоритм показал ускорение S=1,11, четырехпроцессорный – S=1,01. Сокращение длительности арифметических операций не компенсирует рост коммуникационных издержек.

### *3. Параллельно-последовательный алгоритм*

Анализируя структуру Алгоритма 3, укажем, что по мере заполнения вектора неизвестных количество арифметических операций, связанных с модификацией вектора z, сокращается (линейно), а объем пересылаемых данных остается постоянным (вектор из p чисел). В силу этого доля длительности арифметических операций в общем времени вычислений на одном обороте вектора w по кольцу падает с увеличением числа оборотов. Следовательно, на заключительных оборотах по кольцу падает и эффективность распараллеливания.

Предположим, что повысить ускорение можно, разделив алгоритм на две части: параллельную (Алгоритм 3 вплоть до некоторого оборота $r_0$ включительно) и последовательную (Алгоритм 2, начиная с $j=r_0p+1$). При этом для недостаточных значений $r_0$ увеличение ускорения не будет достигаться (за счет существенной доли последовательной части в общем времени вычислений), а для избыточных значений даст знать о себе отмеченный выше недостаток Алгоритма 3.

Переход от параллельного участка алгоритма к последовательному сопровождается пересылкой векторов z от всех процессоров выделенному с их суммированием. Получившийся вектор будет соответствовать z из Алгоритма 2. Вектор правой части $b(r_0p+1:n)$ собирается из соответствующих компонент, хранящихся в локальной памяти всех процессоров вычислительной системы. То же касается матрицы системы.

Представленные в табл. 2,3 и 4 результаты вычислительных экспериментов свидетельствуют о результативности предложенного подхода.

*Таблица 2. Ускорение двухпроцессорного параллельно-последовательного алгоритма для n=10000*

| $r_0$/100 | **5** | **10** | **15** | **20** | **25** | **30** | **35** | **4** | **45** |
|---|---|---|---|---|---|---|---|---|---|
| S | 0,92 | 1,02 | 1,04 | 1,05 | 1,06 | 1,17 | 0,99 | 0,95 | 0,95 |

В случае n=10000 параллельный алгоритм характеризовался отсутствием приемлемого ускорения (S=0,94, табл. 1), применение параллельно-последовательного с параметром $r_0$=3000 позволило увеличить ускорение на 25%, (до S=1,17, табл. 2).

*Таблица 3. Ускорение двухпроцессорного параллельно-последовательного алгоритма для n=12000*

| $r_0$/100 | **10** | **20** | **30** | **40** | **50** |
|---|---|---|---|---|---|
| S | 1,10 | 1,22 | 1,27 | 1,23 | 1,12 |

Для n=12000 использование параллельного алгоритма только начало становиться обоснованным (S=1,04, табл. 1), для параллельно-последовательного с параметром $r_0$=3000 ускорение увеличилось на 22%, (до S=1,27, табл. 3).

*Таблица 4. Ускорение двухпроцессорного параллельно-последовательного алгоритма для n=17000*

| $r_0$/100 | **40** | **50** | **60** | **70** | **80** |
|---|---|---|---|---|---|
| S | 1,37 | 1,43 | 1,43 | 1,38 | 1,26 |

При n=17000 для параллельного алгоритма достигается уже ощутимый результат (S=1,22, табл. 1), использование параллельно-последовательного с параметром $r_0$=5000 характеризуется улучшением этого результата на 17%, (до S=1,43, табл. 3). Приведенное значение ускорения параллельно-

последовательного алгоритма представляется достаточным для организации расчетов, как дающее ощутимый выигрыш от использования двухпроцессорной вычислительной системы.

Отметим также соответствие результатов вычислительных экспериментов предположению о наличии оптимального значения $r_0$, отклонение от которого в большую либо меньшую сторону приводит к снижению ускорения последовательно-параллельного алгоритма.

### *Выводы*

Предложенный в работе параллельно-последовательный алгоритм решения треугольных систем на кольце в силу структуры решаемой задачи характеризуется меньшей длительностью вычислений по сравнению с параллельным. Для приведенных параметров вычислительных экспериментов наблюдался рост ускорения вычислений от 17% до 24%. Предлагаемый подход к синтезу алгоритмов может быть распространен на другие задачи, связанные с матрицами треугольной структуры (например, для решения разреженных СЛАУ большой размерности [8]).

### *Благодарности*

Работа выполнена при поддержке российско-американской программы «Фундаментальные исследования и высшее образование» ("BRHE"), гранта Президента РФ поддержки ведущих научных школ (НШ-3086.2008.9) и гранта РФФИ 07-07-00210а.м.

### *Литература*

1. **Голуб Дж.** Матричные вычисления / Дж. Голуб, Ч. Ван Лоун - М.:Мир, 1999.- 548 с.
2. **Косарев В.И.** 12 лекций по вычислительной математике. Вводный курс / В.И. Косарев - М.:Физматлит, 2000.- 224 с.
3. **Самарский А.А.** Численные методы (Учебное пособие для вузов) / А.А. Самарский, А.В. Гулин - М.:Наука, 1989.- 432 с.
4. Дифракционная компьютерная оптика / под ред. В.А. Сойфера - М.:Физматлит, 2007.-736с.
5. **Li G.** A parallel triangular solver for a distributed-memory multiprocessor / G. Li and T. Coleman // J. Sci and Stat. Comp. 1998.- N9.- p.485-502.
6. **Ортега Д. М.** Введение в параллельные и векторные методы решения линейных систем / Д. М. Ортега - М.: Мир, 1991.-364.
7. **Вальковский В.А.** Элементы параллельного программирования / В.А. Вальковский, В.Е. Котов, А.Г. Марчук, Н.Н. Миренков - М.:Радио и связь, 1983.- 239с.
8. **Ильин В.П.** Проблемы высокопроизводительных технологий решения больших разреженных СЛАУ/ В.П. Ильин// Вычислительные методы и программирование.- 2009.- Т. 10б № 1.- с. 130-136.
9. CUBLAS Library, Published by NVIDIA Corporation 2701 San Tomas Expressway Santa Clara, CA 95050, http://www.nvidia.ru/object/cuda_get_ru.html

### *References*

1. **Golub, G.H.** Matrix Calculations / G.H. Golub, Ch.F. Van Loan – Moscow: Mir, 1999. – 548 p. – (in Russian).
2. **Kosarev V.**I. 12 lectures on calculus mathematics. An introduction course// V.I. Kosarev - Fizmatlit, 2000. – 224 p. – (in Russian).
3. **Samarskiy A.A.** Numerical methods (the Manual for high schools) / A.A. Samarskiy, A.V. Gulin - Moscow: Nauka, 1989.- 432 p. – (in Russian).
4. Methods of Computer Optics (Secondary Edition) / edited by V.A. Soifer – Moscow: Fizmatlit, 2003. – 688 c. – (in Russian).
5. **Li G.** A parallel triangular solver for a distributed-memory multiprocessor / G. Li and T. Coleman // J. Sci and Stat. Comp. 1998.- N9.- p.485-502.
6. **Ortega D.M.** Introduction in parallel and vector methods of the decision of linear systems / D.M. Ortega – Moscow: Mir, 1991. – 364 p. – (in Russian).
7. **Valkovskiy V.A.** Elements of parallel programming/ V.A. Valkovskiy, V.E. Kotov, A.G. Marchuk, N.N. Mirenkov - Moscow: Radio i svyaz, 1983.- 239p. – (in Russian).
8. **Iliin V.P.** Problems of high-efficiency technologies of the decision of the big rarefied systems / V.P. Ilin // Computing methods and programming.- 2009.- V. 10б № 1.- p. 130-136. – (in Russian).
9. CUBLAS Library, Published by NVIDIA Corporation 2701 San Tomas Expressway Santa Clara, CA 95050, http://www.nvidia.ru/object/cuda_get_ru.html

## A PARALLEL-SERIAL ALGORITHM FOR SOLVING TRIANGULAR SYSTEMS ON A PROCESSOR RING

*Dimitry Lvovich Golovashkin[1] (senior researcher, e-mail: dimitriy@smr.ru),*
*Nadezhda Nikolaevna Zhuravleva[2] (graduate student, e-mail: juravleva@mail.ru)*
*[1] Image Processing Systems Institute of the RAS,*
*[2] S. P. Korolyov Samara State Aerospace University*

### *Abstract*

The work is concerned with improving a familiar algorithm for solving triangular systems on a processor ring. With the approach proposed, the low-efficient final part of the algorithm is realized sequentially. The relevance of the technique developed is proved by the results of the computing experiment.

*Key words:* triangular shaped matrix, parallel algorithm, processor ring.

*В редакцию поступила 01.06.2009г.*